SUBSTITUTE SPECIFICATION – "CLEAN" VERSION

APPARATUS AND METHOD FOR SUPPLYING ELECTRONIC CONTENT TO NETWORK APPLIANCES

Inventors: Jack B. Strong, John N. Lehner, Jonathan J. Kleid, and Vivek Patel

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is based on and claims priority under 35 USC § 119(e) of U.S. Provisional Patent Application No. 60/212,147, filed on June 16, 2000 and entitled "Apparatus and Method for Supplying Electronic Content to Network Appliances", which is incorporated by reference herein.

COPYRIGHT NOTICE

BACKGROUND

[0003] Until recently, the vast majority of internet users accessed the World Wide Web via personal computers and workstations. Web designers could safely assume that the devices accessing their sites would have similar display capabilities, processor speed,

and connection bandwidth. With careful design, it was possible to produce a single version of a web site that displayed satisfactorily in nearly all browsers. Unfortunately, this is no longer possible. The number and variety of so-called "network appliances" has exploded. Millions of internet users now log on via set top boxes, personal digital assistants (PDAs), and cellular telephones. Analysts have predicted that more people will soon access the web via network appliances than via personal computers. Now, web designers must take into account that browsers accessing their sites may display just a few words of text at a time, or they may display hundreds. Other browsers have no display at all and instead read the content to the user through a telephone. Processor speeds and bandwidth may vary by an order of magnitude. To make matters worse, different devices support different sets of file formats, often with no intersection at all.

[0004] Web content providers that need to support users on all these devices typically have to provide multiple versions of their sites. Amazon.com currently has three versions of its site: the conventional HTML version; an HDML version for display on first generation cell phone browsers; and a "web clipping" version for certain handheld computing devices. A fourth version will be for WML capable cell phones. Only content providers with the deepest pockets can afford to author so many different versions of their sites. Even for giants like Amazon.com, this approach is clearly not scalable as the variety of network appliances continues to increase.

[0005] As with many problems in computer science, this dilemma can be at least partially solved by adding a level of indirection. In internet terms, the level of indirection between a browser and a web server is called a proxy server. The model is simple. When

a browser needs to retrieve a document from the web, it sends a request to the proxy server using the browser's native protocol. The proxy then retrieves the document from the server. If the document happens to be in a format that the browser is capable of displaying, the proxy will simply forward it along. This is where the process ends for most, if not all, traditional proxy servers.

SUMMARY

[0006] Embodiments of the present invention relate to a system and method for communicating data from a web content provider to a wireless computing device (e.g., a PDA, a cellular phone) via a proxy server. The proxy server transforms web content "on-the-fly" into a streamlined format, optimizes it for display, and delivers it to the device, where it is progressively rendered by a browser installed on the device.

[0007] The features and advantages described herein are not all inclusive, and, in particular, many additional features and advantages will be apparent to those skilled in the art in view of the following description. Moreover, it should be noted that the language used herein has been principally selected for readability and instructional purposes and may not have been selected to circumscribe the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Fig. 1 is a simplified diagrammatic representation of an embodiment of the invention.

[0009] Fig. 2 is a simplified illustration of a progressive rendering technique employed in an embodiment of the invention.

[0010] Each of the figures referenced above depict an embodiment of the present invention for purposes of illustration only. Those skilled in the art will readily recognize from the following description that one or more other embodiments of the structures, methods, and systems illustrated herein may be used without departing from the principles of the present invention.

DETAILED DESCRIPTION

[0011] In the following description of embodiments of the present invention, numerous specific details are set forth in order to provide a more thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without one or more of these specific details. In other instances, well-known features have not been described in detail to avoid unnecessarily complicating the description.

[0012] In general, embodiments of the present invention relate to a system and method for communicating data from a web content provider to a wireless computing device (e.g., a PDA, a cellular phone) via a proxy server. The proxy server transforms web content "on-the-fly" into a streamlined format, optimizes it for display, and delivers it to the device, where it is progressively rendered by a browser installed on the device.

[0013] When a document (e.g., a web page) received by the proxy server is not in a

format capable of being displayed by a requesting wireless device, the proxy server will attempt to transform the document into an acceptable format. For example, if the web content provider sends a document marked up in HTML, and the browser of the requesting wireless device only supports WML, the proxy server can run the document through an HTML-to-WML translation module. Other formats possibly supported by wireless devices to which the proxy server may transform format include, for example, HDML, PQA (also known as the "web clipping" format), XHTML, Basic, and AvantGo's non-standard HTML subset.

[0014] Further, in one or more embodiments, this model can be extended beyond simply file format translations to encompass display optimizations as well. For example, if the web content provider sends a 256-color GIF image, and the requesting wireless device supports only 4 shades of gray, the proxy server may then reduce the bit depth of the image, thereby (i) saving the wireless device the processing overhead of doing the conversion itself and (ii) reducing the file size to optimize transmission speed over a low bandwidth connection.

[0015] Moreover, those skilled in the art will appreciate that the conversion process is transparent to both the web content developer and the user of browser of the requesting wireless device. In other words, a proxy server in accordance with one or more embodiments frees end users (both content producers and content consumers) from concerning themselves with the proxy server. Typical prior art proxy servers require either the content provider or browser developer to maintain the proxy server. This included not only maintaining a server farm, but also keeping the transformation software

up-to-date on all of the latest file formats and device capabilities. In one or more embodiments, by disassociating the transformation engine from both the server and the client, a third party application service provider may maintain the proxy. Further, the transformation software may be upgraded to support new devices and formats in a manner that is transparent to both the content provider and the end user. This degree of transparency is believed to be important to the widespread adoption of proxy-based transformation technology. Moreover, in one or more embodiments, the transformation software may be updated automatically via, for example, the Internet.

[0016] The transformation of content by a proxy server in accordance with one or more embodiments may be dynamic and/or based on user-agent profiles stored locally on the proxy server, provided by the user-agent, and/or retrieved from another server based in information provided by the user-agent. The profiles may specify supported content types, display characteristics of the device, and/or transformation preferences, and allow the proxy server to optimize a display for user-agents whose capabilities are not known at the time the proxy server is deployed.

[0017] Now referring to Figure 1, prior art proxy servers **12** (shown as "other systems") wait for an entire web page to be received from a web content provider **10**. Once the data for the entire web page has been received, the proxy server **12** transforms the data to a format for display by a web browser **14** of a requesting wireless device **20**. The web browser **14** then waits for the entire web page data to be received prior to displaying the web page.

[0018] Still referring to Figure 1, in one or more embodiments, a proxy server **16**

receives web page data in a streamlined manner and transforms received web page data "on-the-fly". In other words, the proxy server **16** transforms and sends for display by a web browser **18** web page data as the proxy server **16** receives web page data from the web content provider **10**. Thus, the proxy server **16** may receive portions of data for a web page after other portions of the web page data have already been transformed and sent by the proxy server **16**. Moreover, the web browser **18** in accordance with one or more embodiments is arranged to progressively render received web page data. In other words, the web browser **18** may display the web page data as it receives it; it does not have to wait for all of the web page data to be received.

[0019] Thus, the proxy server **16** transforms web content into a streamlined format and optimizes it for display on a screen of a requesting wireless device **20**. The requesting wireless device **20** may have a web browser **18** that is arranged to progressively render received web page data. In such a manner, users may view content created in any format through a single web browser (instead of having to use separate applications) (see Figure 2), and do so in an environment in which web pages load relatively quickly due to the content streamlining mechanism described with reference to Figure 1.

[0020] Accordingly, a system in accordance with one or more embodiments involves at least one of on-the-fly transformation, on-the-fly content transformation for optimal display on a device screen, "rapid streaming" technology, and a web browser that progressively renders received web page data.

[0021] A web browser in accordance with one or more embodiments will now be

described. The web browser application may be structured as an object-oriented program. One set of classes may be primarily focused on aspects of a particular computing environment, such as responding to launch codes and user events. A second set of classes may be responsible for browser specific tasks, such as creating a parse tree from a WML document and displaying the data. A third set may be responsible for interfacing between OS independent browsing code and operating system code. A fourth set may include generic, reusable components such as vectors and iterators. Sub-goals may be to create a reusable framework for application development, as well as a platform independent browser kernel. Those skilled in the art will note that these sub-goals, among others, may naturally leas to clean abstractions.

Application

- Top-most level of application
- Determines whether the current operating environment is suitable for this application (i.e. make sure the OS version number is not too old).
- Handles launch codes (e.g. normal launch, soft reset, beam received).
- Receives all user input; must dispatch commands to the appropriate handlers.
- An application is defined by a series of forms, so this class must manage a set of forms that are each active at different times.

Form

- Represents abstract form (e.g., a window). Meant to be sub-classed by forms with specific functions.

- Supplies UI functions common to all (or at least most) forms.
- Contains zero or more views, arranged in some form specific way.

View

- Represents abstract region for displaying some sort of content or UI.
- Not nestable (i.e. a view cannot contain another view)
- Gives forms more flexibility since views are location independent (i.e. all drawing within a view is relative to the view's coordinates)

Preferences

- Manages the storage of persistent preferences used by the application.

EventHandler

- Abstract interface for any component that can handle user events (i.e. Form, View)

Browser Components

Expat

- Collection of open source C files which parse XML files.
- Expat as a shared library to perform parsing, and will subsequently build a parse tree based on its output.

XMLParser

- Wrapper around expat and the parse tree (expat could be replaced by a different parser, and only this class would be affected).

- Manages the construction of the tree, using expat to build a tree composed of XMLElements and XMLAttributes.

XMLElement

- Single XML tag, with attributes and data

XMLAttribute

- Represents a single XML attribute

WMLGlyph

- Graphical representation of XMLElement.
- Based on recursive composition technique in *Design Patterns,* Chapter 1 (by the Gang of Four).
- A glyph can be a container for other glyphs, or can be a leaf.
- Glyphs draw themselves, and handle UI tasks within its boundaries.
- Examples of glyphs are images, blocks of text, and lines of text.

HTTPProtocol

- Implements HTTP Protocol for any underlying session mechanism (i.e. TCP)
- Possibly composed of public domain source code.

PalmOS/WML Browser Glue Components

WMLForm

- Subclass of Form, defines browser specific form features, such as back and forward arrows.
- Contains a WMLView

WMLView

- Displays WML content. Can be embedded in any application that implements the Form and View classes described above.
- Gives WMLGlyph a display area for layout
- Responsible for scrolling WML content

WMLSession

- Responsible for retrieving data, and supplying pages on demand to the WML browser.
- The underlying mechanisms will be pluggable, meaning regardless of whether a modem is used, or a Bluetooth module communicating to a cell phone, the rest of the browser will work in the same way.

TCPSession

- Subclass of WMLSession which uses built-in TCP stack
- Provides socket level instructions to enable HTTP

Generic Classes

Vector

- Easy to use data structure for managing groups of data

Iterator

- Iterator for vector, or any other class which represents a collection of objects

Types

- Defines commonly used types such as Point and Rectangle

Component Interface Design

Application

- *main:* entry point for application. Calls appropriate function depending on the launch code passed in. Performs suitability checks on environment.
- *eventLoop:* dispatches commands to appropriate eventHandlers, which have registered themselves. Keeps track of current handler.
- *registerHandler:* called by forms that can handle events, to tell the application that they should be used with forms of certain ID's. This is necessary because in the PalmOS, forms are labeled with ID numbers. By registering themselves, form objects become associated with the notion of a form.
- *launchNormal:* virtual function called by main when program is launched normally. Can be subclassed by applications to customize behavior.

Form

- *formLoad, formOpen, formClose:* Called by main event loop when this object is the current handler. formLoad should load resources needed to display the form, without writing anything to the screen. formOpen should draw the form itself. formClose should erase the form and release any resources.
- *draw:* Called by eventLoop (and possibly other functions) when the form's contents should be drawn/redrawn.
- *addView.* Adds new view to the form.
- *handleEvent:* Processes form specific events

View

- *handleEvent:* Processes view specific events
- *getBounds* Returns the area used by the view

Preferences

- *read:* reads prefs from storage into temporary data structure
- *write:* writes prefs from temporary data structure into storage
- Subclasses should provide assessors/modifiers for specific preferences.

EventHandler

- *doesHandleType:* Returns true if this object handles the specified event type (i.e. menu action)
- *doesHandleObject:* Returns true if this object handles the specified object (i.e. form with ID #4300)
- *handleEvent:* Actually handle the event. (Must be overridden by subclass)

Expat

- *XML_SetElementHandler:* Used to set the callback for new elements. Callback is called for each XML tag seen.
- *XML_SetCharacterDataHandler:* Used to set callback for data handler.

XMLParser

- *parse:* parses specified document, returns root element of document (of type XMLElement)

XMLElement

- *addAttribute:* adds attribute to element
- *getAttribute:* returns vector of attributes
- *getTag:* returns tag name

XMLAttribute

- assessor/modifier for name, value pair

WMLGlyph

- *intersects:* returns true if this glyph intersects specified point
- *insert:* inserts a sub-glyph into this glyph
- *activate:* called when the UI element containing the glyph is activated
- *deactivate:* called when the UI element containing the glyph is deactivated
- *draw:* recursively draws glyph, and all its sub-glyphs
- *getExtent:* returns bounds of glyph
- *parent:* returns parent glyph
- *children:* returns vector of children glyphs, if any **HTTPProtocol**
- *sendRequest:* sends request to server, returns requested data. Uses underlying session protocol as provided on device.

WMLForm

- overrides functions of Form, adding appropriate behavior **WMLView**
- overrides functions of View, adding appropriate behavior **WMLSession**
- *set View:* way for view to register itself with a session.
- *goTo:* called by view to get data of passed in URL

TCPSession

- *goTo:* overridden from WMLSession
- provides underlying layer for HTTPProtocol to interact with, such as open socket, and read/write data.

Vector

- *add:* adds element to end of vector
- *insert:* inserts element into some position in vector
- *remove:* removes specified element
- *removeAll:* Empties vector
- *getElementAt:* returns specified element
- *getNumElements:* returns count of all elements.

Iterator

- *hasNext:* returns true if there are more elements
- *getNext:* returns next element

Data Structure Design

[0022] Important data structures are the representations of the WML document. After parsing, a simple parse tree is created. The parse tree's structure may be based on the notion that a child node is nested within their parent in the corresponding XML. For example, if a run of text looks like:

* text *

then the element representing `i' will be a child of the element representing `p'. Further,

the text element will be a child of i. XML attributes are represented as a vector of string pairs, where each pair corresponds to a name and value.

[0023] This parse tree is independent from the glyph tree, although the two trees are similar in structure. The glyph tree has at least one glyph per parse tree element, and possibly more. For example, a block of text that spans multiple lines is represented as a single element in the parse tree. In the glyph tree, however, it will become multiple glyphs — one for each line, and one for the text block itself. This is based on the Composite pattern, described in *Design Patterns.*

[0024] Vectors are used extensively as a convenient abstraction on top of arrays. STL may be used for data structures such as this. Data structures may be optimized in view of specific constraints faced.

Algorithm Design

[0025] The crux of the browser is the layout algorithm. It should be able to flow a wide variety of objects, and store their position in order to respond to user input (i.e. clicking on a link). It should be flexible enough to work concurrently with data download, so it must operate incrementally. It should be efficient, as computing resources may be limited. To facilitate these needs, the glyph tree to represent on screen information may be used, as described in the previous section. The glyph tree is flexible, in that a block of text can be reformatted into its constituent lines at any point and redrawn, if for example an image is newly downloaded and changes the flow of the document.

[0026] The glyph draw function may be implemented in a relatively coordinate

independent way. This means that a glyph may be passed a display area in which to draw, which it can treat as its own canvas. Each glyph may also calculate the sub-areas available to its sub-glyphs. The rendering algorithm is then composed of all glyph types following this general contract. As long as glyphs draw only in the space allotted, and gives their sub-glyphs adequate screen real estate, then the contract is fulfilled. Because of the recursive nature of the algorithm, considerable information can be cached. For example, the extent of a glyph can be stored, so the getExtent function will not need to examine all sub-glyphs to recompute the extent.

[0027] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of the above description, will appreciate that other embodiments may be devised which do not depart from the scope of the present invention as described herein. Accordingly, the scope of the present invention should be limited only by the appended claims.

CLAIMS

What is claimed is:

1. A method for providing multimedia data over an electronic network to a wireless computing device having a browsing program, comprising:

receiving a first portion of multimedia data having a first display format at a proxy server, wherein the first portion has text data and graphical data;

converting the first display format to a second display format, wherein the browsing program interprets multimedia data in the second display format; and

transmitting the text data over the electronic network to the wireless computing device before transmitting the graphical data and before receiving a second portion of multimedia data.